# **Звіт до курсової роботи**

з дисципліни «бази даних та інформаційні системи»

Виконав: студент гр.

ПМ-13-1

Кривоносов О. Д.

Перевірив:

Мащенко Л. В.

ЗМІСТ

[**Звіт до курсової роботи** 0](#_Toc467811253)

[Вступ 3](#_Toc467811254)

[1. Проектування системи баз даних 3](#_Toc467811255)

[1.1 Постановка задачі 3](#_Toc467811256)

[1.2 Основні задачі, що треба розв'язати 3](#_Toc467811257)

[1.3. Вхідні та вихідні документи, звіти 4](#_Toc467811258)

[1.4. Універсальна таблиця, виявлення аномалій 4](#_Toc467811259)

[1.5. Виділення об’єктів 4](#_Toc467811260)

[1.6. Функціональні залежності 5](#_Toc467811261)

[1.7. Нормалізація відношень та її обґрунтування 6](#_Toc467811262)

[1.8. Дослідження запитальних зв’язків 7](#_Toc467811263)

[1.9. Встановлення структурних зв’язків 10](#_Toc467811264)

[1.10. Побудова граф-схеми інфологічної моделі та перевірка її коректності 10](#_Toc467811265)

[2. Забезпечення основних характеристик систем баз даних 11](#_Toc467811266)

[3. Реалізація системи баз даних 11](#_Toc467811267)

[3.1. Обґрунтування вибору СУБД 11](#_Toc467811268)

[3.2. Можливість використання системи в режимі мережі 11](#_Toc467811269)

[3.3. Опис та структурна схема програми 11](#_Toc467811270)

[3.4. Середовище розробки інтерфейсу та його опис 11](#_Toc467811271)

[3.5. Призначення процедур та функцій 12](#_Toc467811272)

[3.6. Застосування мови SQL для створення запитів 12](#_Toc467811273)

[3.7. Створення та використання класів, тригерів, процедур 12](#_Toc467811274)

[4. Безпека та таємність системи 12](#_Toc467811275)

[4.1. Ведення журналу роботи системи 12](#_Toc467811276)

[4.2. Процедури адміністратора 12](#_Toc467811277)

[5 Інструкції по використанню та супроводженню 12](#_Toc467811278)

[5.1. Інструкція адміністратору 12](#_Toc467811279)

[5.2. Інструкція програмісту 12](#_Toc467811280)

[5.3. Інструкція користувачу 12](#_Toc467811281)

[6. Тестування та налагодження 12](#_Toc467811282)

[6.1. Підготовка тестів для різних етапів функціонування системи та їх реалізація 12](#_Toc467811283)

[6.2. Опис результатів тестування та налагодження 12](#_Toc467811284)

[Висновки 13](#_Toc467811285)

[СПИСОК ЛІТЕРАТУРИ 13](#_Toc467811286)

[ДОДАТКИ 13](#_Toc467811287)

[ДОДАТОК (код програми) 13](#_Toc467811288)

# Вступ

# Проектування системи баз даних

## 1.1 Постановка задачі

Т е м а. Чат (програма, яка забезпечує передавання текстових повідомлень).

Ц i л ь. Ознайомитися з мовою SQL для сервера MYSQL, API MYSQL для мови C, API WINSOCK у c++, побудувати деякі обгортки та GUARDS для автоматизації звільнення ресурсів для MYSQL та WINSOCK дескрипторів; створити повноцінний чат (аналог додатку “Skype”).

*Постановка завдання*

Реалізувати чат, який матиме такий архітектурний вигляд:  
 інтерфейс ------- сервер менеджер чату ------- сервер бази даних.

Виконати такі умови:

1. Всі повідомлення мають зберігатися на сервері бази даних.
2. Користувач може мати друзів, може подати заявку в друзі, може прийняти заявку в друзі, може переглянути, чи є інший користувач у мережі, може пересвідчитись, чи були прочитані їм повідомлення від конкретного друга.
3. Користувач може шукати нових друзів серед користувачів, чатів, друзів, друзів друзів за ім’ям або логіном, може виконувати пошук чатів по даті повідомлень, які присутні у чаті.
4. Користувач може створювати чати, додавати члевів чату і встановлювати рівні доступу чденам чату.
5. Користувачів не більше ніж 2000000, і не більше 40000 користувачів одночасно.

## 1.2 Основні задачі, що треба розв'язати

1. Прочитати такі книжки:
   1. MYSQL\_Кузнецов М.В., Симдянов И.В. - MySQL 5 (в подлиннике) – 2010
   2. У. Р. Стивенс - UNIX. Разработка сетевых приложений, 3-е изд. – 2007
   3. Entoni\_Uilyams\_-\_Parallelnoe\_programmirovanie
2. Ознайомитись з mysql API мови C за допомоги ресурсів:
   1. <http://www.mysql.ru/docs/man/C.html>
   2. <http://zetcode.com/db/mysqlc/>
3. Забезпечити змогу створювати конференції так, щоб користувач, що зробив конференцію, міг редагувати її параметри : додавати членів чату, надавати їм один з рівнів доступу (привілейований користувач – може додавати нових користувачів, змінювати назву та аватар чату, звичайний користувач – може читати і писати повідомлення, без доступу – не може писати повідомлення, але може читати повідомлення, котрі були створені за часу, коли користувач був членом чату з вищім рівнем доступу).
4. Розробити базу даних:  
   На сервері бази даних мають бути збережені процедури та тригери, завдяки яким спрощується реалізація запитів до СУБД і зменшується трафік між сервером чату і сервером БД.
5. Розробити сервер обслуговування клієнтів.
   1. GUARD WinSock
   2. GUARD mysql
   3. Interrupted thread
   4. Thread pool
   5. Data serialization
   6. Query processing
6. Розробити інтерфейс.

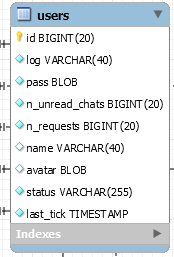
## 1.3. Вхідні та вихідні документи, звіти

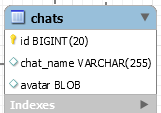
?

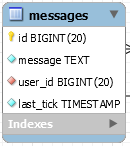
## 1.4. Універсальна таблиця, виявлення аномалій

?

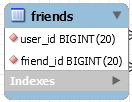
## 1.5. Виділення об’єктів

Користувачі – таблиця users.  


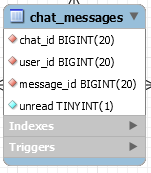
Чати – таблиця chats.  


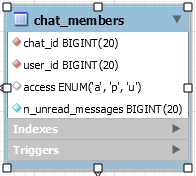
Повідомлення – таблиця messages, має зовнішній ключ на користувача, що створив повідомлення, залежність – один до одного (messages1 - 1users).  


## 1.6. Функціональні залежності

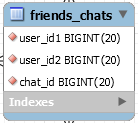
Друзі – таблиця friends. (users∞ - ∞users)  


Запит в друзі – таблиця requests. (users∞ - ∞users)  


Повідомлення чату – таблиця chat\_messages. ((chats1 - 1messages)∞ - ∞users)  


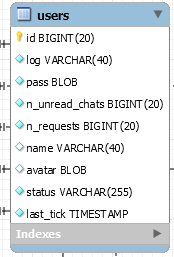
Члени чатів, таблиця chat\_members. (chats∞ - ∞users)  


Чати друзів – таблиця friends\_chats.

Коли створюється пара друзів в таблиці friends – створюється й чат для переписки між друзями (якщо такого чату ще не було), де вони (користувачі-друзі) отримують рівень доступу ‘u’ (звичайний користувач чату). В таблицю friends\_chats додається запис **[**min(userID1, userID2), max(userID1, userID2), chatID1**]**.  
Записи з цієї таблиці не видаляються (не потрібно видаляти), бо якщо один з друзів видалить іншого зі свого списку друзів, повідомлення між друзями не видаляються. Якщо ці користувачі знов стануть друзями – для них не створюється новий чат, а використовується вже існуючий, вказник на який зберігався в таблиці friends\_chats.  
((users∞ - ∞users)1 - 1chats)  


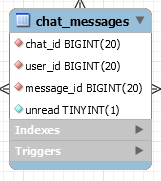
## 1.7. Нормалізація відношень та її обґрунтування

Розглянемо таблицю chat\_messages:

.

Таблиця в 1НФ, але не в 2НФ, бо не виконується умова невиводимості. Первинним ключем є id, але, при наявності унікального для кожного користувача поля log, усі інші поля можна вивести за значенням log. Навіщо залишати поле id, якщо log однозначно визначає користувача? Це зроблено для більш зручного користування базою даних, бо користувацьким програмам простіше оперувати 4-х байтним числом ідентифікатором, а не 40-ка байтною строкою.

Розглянемо таблицю chat\_messages:

.

Ця таблиця одночасно виконує роль декількох об’єктів зв’язок: показує яке повідомлення належить до якого чату (chat\_id, message\_id), показує яке повідомлення було прочитане конкретним користувачем (user\_id, message\_id, unread).

Таблиця в 1НФ, але не в 2НФ, бо не виконується умова невиводимості: окреме значення message\_id визначає поле chat\_id і таких записів в таблиці, де message\_id визначає поле chat\_id буде стільки, скільки користувачів отримало повідомлення message\_id. При наявності необхідності запитів, де окремо треба визначати до якого чату відноситься повідомлення цю таблицю одразу необхідно поділити на дві, але такої необхідності нема. Навпаки з тими запитами до цієї таблиці, що існують: GET\_MESSAGES, FIND\_CHAT\_BY\_TIME\_RANGE, простіше тримати поля (chat\_id, user\_id, message\_id, unread) разом, бо якщо б вони були в різних таблицях, виникала б необхідність для кожного разу будувати таблицю, що з’єдную ці поля разом.

## 1.8. Дослідження запитальних зв’язків

Розглянемо декілька запитальних запитів, що можуть поставити під сумнів правильність зв’язків між таблицями:

1. Які користувачі мають ім’я схоже на задане?
2. Які друзі заданого користувача мають ім’я схоже на задане?
3. Які чати користувача мають ім’я схоже на задане?
4. В яких чатах користувача є повідомлення, за заданий проміжок часу?
5. Які в заданих користувачів спільні друзі?

Словесно відповісти на ці питання можна так:

1. Оскільки в нас є список користувачів (users), а користувачі мають ім’я, будуємо список користувачів, що мають ім’я схоже на задане.
2. Оскільки в нас є список уіх пар друзів (friends), будуємо список друзів заданого користувача.  
   Оскільки друзі – це користувачі, а користувачі мають ім’я, будуємо список друзів, що мають ім’я схоже на задане.
3. Оскільки в нас є список усіх членів усіх чатів (chat\_members), будуємо список чатів, для яких заданий користувач є членом чату.  
   Оскільки чати мають ім’я, будуємо список чатів, що мають ім’я схоже на задане.
4. Оскільки в нас є список усіх членів усіх чатів (chat\_members), будуємо список чатів, для яких заданий користувач є членом чату.  
   Оскільки в нас є список усіх повідомлень усіх чатів, надісланих конкретному користувачеві (chat\_messages), для кожного чату будуємо список повідомлень, надісланих заданому користувачеві.  
   Оскільки в нас є список усіх повідомлень (chat\_members), а для кожного повідомлення є час його створення, для кожного чату вибераємо такі повідомлення, щ обули створені в заданий проміжок часу.
5. Оскільки в нас є список уіх пар друзів (friends), будуємо списки друзів для кожного з заданих користувачів.  
   Для перших двох списків вибераємо тіх друзів, що є і в першому, і в другому.  
   Дня отриманого списку і наступного скиску вибераємо тіх друзів, що є і в першому, і в другому.  
   …  
   Таким чином ми отримаємо список усіх спільних друзів.

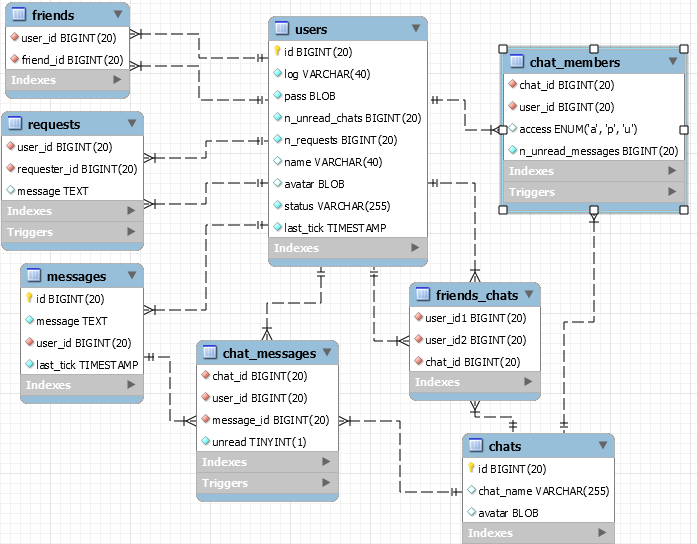
Можна бачити, що усі обєкти пов’язані між собою так, що на питання поставлене про один об’єкт, можна відповісти виходячи з відомостей про інший об’єкт.

## 1.9. Встановлення структурних зв’язків

?

Уже ведь описано все в 1.6. Функціональні залежності, то ту тогда писать, или что там тогда писать.

## 1.10. Побудова граф-схеми інфологічної моделі та перевірка її коректності



(Рис.1. Граф-схема інфологічної моделі)

Є три типи об’єктів : users, chats, messages, які мають первинний ключ, тому всі зв’язки можуть бути описані лише за допомогою зовнішніх ключів до первинних ключів одного з типів об’єктів users, chats, messages. На показаній граф-схемі ця умова виконана.

Усі зв’язки відповідають тим, що описані в пунктах [1.5](#_1.5._Виділення_об’єктів) і [1.6](#_1.6._Функціональні_залежності).

Покажемо на граф-схемі шляхи обходу таблиць, які використовуються відповідми на запитальні запити в [1.8](#_1.8._Дослідження_запитальних).

1. Які користувачі мають ім’я схоже на задане?  
   **[users:** user\_name => (user\_name ⬄ user\_id) => user\_id**]**
2. Які друзі заданого користувача мають ім’я схоже на задане?  
   **[friends:** user\_id => (user\_id ⬄ friend\_id) => friend\_id**]** ------> **[users:** id => (id ⬄ name) => name**]**
3. Які чати користувача мають ім’я схоже на задане?  
   **[ chat\_members:** user\_id => (user\_id ⬄ chat\_id) => chat\_id**]** ------> **[chats:** id => (id ⬄ chat\_name) => chat\_name**]**
4. В яких чатах користувача є повідомлення, за заданий прміжок часу? **[ chat\_members:** user\_id => (user\_id ⬄ chat\_id) => (chat\_id, user\_id)**]** ------>  
   ------> **[chat\_messages:** (chat\_id, user\_id) => message\_id**]** ------> **[messages:** id => (id ⬄ last\_tick) => last\_tick**]**
5. Які в заданих користувачів спільні друзі?  
   **[friends:** user\_id => (user\_id ⬄ friend\_id) => friend\_id**]**

# 2. Забезпечення основних характеристик систем баз даних

База даних створювалася з розрахунком на те, що об’єкти (записи з первинним ключем) не видалялися і їх первинні ключі не змінювалися; видалятися можуть лише зв’язки. Виходячи з цього, для зовнішніх ключів непотрібно вказувати жодних ONUPDATE, ONDELETE. Тому усі питання про цілісність данних переносяться на тригери таблиць та запити.

Треба виконати такі умови цілісності та несуперечливісті:

1. Додавання в друзі може виконуватися тільки створенням запиту в друзі одним користувачем і прийняттям запиту іншим користувачем.
2. Для кожного нового відношення «друзі» в таблиці friends створювати два симетричних записи (friend\_id1, friend\_id2), (friend\_id2, friend\_id1); якщо ще немає чата друзів – створити його.
3. Для кожного видалення з друзів – видаляти обидві створені записи в таблиці friends.
4. Для кожного додавання запиту в друзі треба перевіряти, чи є задані користувачу вже друзями.
5. Для кожного нового відношення «запит в друзі» збільшувати поле users.n\_requests на 1.
6. Для кожного відхилення або прийняття в друзі зменшувати поле users.n\_requests на 1.
7. Повідомлення можуть надсилатися тільки в чат (не може існувати повідомлення без чату).  
   Для виконання цієї умови доцільно було б встановити стовбець в таблиці messages, але для виконання запитів зручніше використовувати таблицю chat\_messages, тому таке поле в messages було б невикористаним (витрачало б ресурси системи марно).
8. Для кожного додавання повідомлення в чат треба перевіряти, чи є заданий користувач членом чату, та чи має він відповідний рівень доступу.
9. Для кожного нового повідомлення створювати список користувачів, що отримали це повідомлення (членів чату, які на момент створення повідомлення мали рівень доступу вищий за рівень «без доступу»).
10. Для кожного нового відношення «повідомлення користівачеві в чаті» в таблиці chat\_messages встановлювати поле chat\_messages.unread=TRUE і збільшувати поле chat\_members.n\_unread\_messages для відповідного користувача (члена чату) на 1.
11. Для кожного разу, коли в деякому записі таблиці chat\_messages поле chat\_messages.unread було TRUE, а стало FALSE, зменшувати поле chat\_members.n\_unread\_messages для відповідного користувача (члена чату) на 1.
12. Для кожного разу, коли в деякому записі таблиці chat\_members поле chat\_members.n\_unread\_messages було 0, а стало 1, збільшувати поле users.n\_unread\_chats для відповідного користувача (члена чату) на 1.
13. Для кожного разу, коли в деякому записі таблиці chat\_members поле chat\_members.n\_unread\_messages було 1, а стало 0, зменшувати поле users.n\_unread\_chats для відповідного користувача (члена чату) на 1.

Масштабованість такої інформційної системі відносно кількості користувачі лінійна – негко масштабується, а от відносно кількості повідомлень не лінійна (складніше) – на кожне повідомтення в таблиці chat\_messages з’являється стільки записів, скільки кооистувачів отримали повідомлення.

Масштабованість такої інформційної системі відносно розширення функціоналу задовільна, бо легко встановити нові відношення, ввести нові контекстні поля в таблиці об’єктів або об’єктів-зв’язок, ввести нові об’єкти і зв’язати їх з існуючими.

# 3. Реалізація системи баз даних

## 3.1. Обґрунтування вибору СУБД

## 3.2. Можливість використання системи в режимі мережі

## 3.3. Опис та структурна схема програми

## 3.4. Середовище розробки інтерфейсу та його опис

## 3.5. Призначення процедур та функцій

## 3.6. Застосування мови SQL для створення запитів

## 3.7. Створення та використання класів, тригерів, процедур

* FIND\_CHAT(userID, regular\_chat\_name VARCHAR(255), offset\_, number).  
  Цей запит виконує пошук чатів користувача, в яких ім’я задовольняє регулярному виразу regular\_chat\_name.

Опис FIND\_CHAT:

DROP PROCEDURE IF EXISTS FIND\_CHAT//

CREATE PROCEDURE FIND\_CHAT(userID BIGINT UNSIGNED, regular\_chat\_name VARCHAR(255), offset\_ BIGINT UNSIGNED, number BIGINT UNSIGNED)

BEGIN

START TRANSACTION;

SELECT

my\_chat.chats.id,

my\_chat.chats.chat\_name,

my\_chat.chats.avatar,

my\_chat.chat\_members.access,

my\_chat.chat\_members.n\_unread\_messages

FROM my\_chat.chat\_members, my\_chat.chats

WHERE my\_chat.chat\_members.user\_id = userID AND my\_chat.chat\_members.chat\_id = my\_chat.chats.id

AND (my\_chat.chats.chat\_name REGEXP regular\_chat\_name)

LIMIT offset\_, number;

COMMIT;

END//

* FIND\_CHAT\_BY\_TIME\_RANGE(  
   userID BIGINT UNSIGNED,  
   UNIX\_TIMESTAMP\_left\_end BIGINT UNSIGNED,  
   UNIX\_TIMESTAMP\_right\_end BIGINT UNSIGNED,  
   offset\_ BIGINT UNSIGNED,  
   number BIGINT UNSIGNED  
  ).  
  Запит більш складний за FIND\_CHAT, виконує пошук чатів користувача, в яких є повідомлення, що створені в проміжок часу   
  [UNIX\_TIMESTAMP\_left\_end, UNIX\_TIMESTAMP\_right\_end].

Опис FIND\_CHAT\_BY\_TIME\_RANGE:  
  
DROP PROCEDURE IF EXISTS FIND\_CHAT\_BY\_TIME\_RANGE//

CREATE PROCEDURE FIND\_CHAT\_BY\_TIME\_RANGE(userID BIGINT UNSIGNED, UNIX\_TIMESTAMP\_left\_end BIGINT UNSIGNED, UNIX\_TIMESTAMP\_right\_end BIGINT UNSIGNED, offset\_ BIGINT UNSIGNED, number BIGINT UNSIGNED)

BEGIN

SET @left = LEAST(UNIX\_TIMESTAMP\_left\_end, UNIX\_TIMESTAMP\_right\_end);

SET @right = GREATEST(UNIX\_TIMESTAMP\_left\_end, UNIX\_TIMESTAMP\_right\_end);

START TRANSACTION;

SELECT

my\_chat.chats.id,

my\_chat.chats.chat\_name,

my\_chat.chats.avatar,

my\_chat.chat\_members.access,

my\_chat.chat\_members.n\_unread\_messages

FROM my\_chat.chat\_members, my\_chat.chats

WHERE my\_chat.chat\_members.user\_id = userID

AND (

SELECT COUNT(\*) FROM my\_chat.chat\_messages

WHERE

my\_chat.chat\_messages.chat\_id = my\_chat.chat\_members.chat\_id

AND

my\_chat.chat\_messages.user\_id = my\_chat.chat\_members.user\_id

AND

(

SELECT COUNT(my\_chat.messages.id) FROM my\_chat.messages

WHERE

my\_chat.chat\_messages.message\_id = my\_chat.messages.id

AND

UNIX\_TIMESTAMP(my\_chat.messages.last\_tick) BETWEEN @left AND @right

) > 0

LIMIT 1

) > 0

AND my\_chat.chat\_members.chat\_id = my\_chat.chats.id

LIMIT offset\_, number;

COMMIT;

END//

# 4. Безпека та таємність системи

## 4.1. Ведення журналу роботи системи

## 4.2. Процедури адміністратора

# 5 Інструкції по використанню та супроводженню

## 5.1. Інструкція адміністратору

## 5.2. Інструкція програмісту

## 5.3. Інструкція користувачу

## 6. Тестування та налагодження

## 6.1. Підготовка тестів для різних етапів функціонування системи та їх реалізація

## 6.2. Опис результатів тестування та налагодження

# Висновки

# СПИСОК ЛІТЕРАТУРИ

1. <http://www.mysql.ru/docs/man/C.html>
2. <http://zetcode.com/db/mysqlc/>
3. «MYSQL Кузнецов М.В., Симдянов И.В. - MySQL 5 (в подлиннике) – 2010»
4. «У. Р. Стивенс - UNIX. Разработка сетевых приложений, 3-е изд. - 2007»
5. «Entoni Uilyams – Параллельное программирование на C++ в действии,  
   1-е изд. - 2012»

# ДОДАТКИ

## ДОДАТОК (код програми)